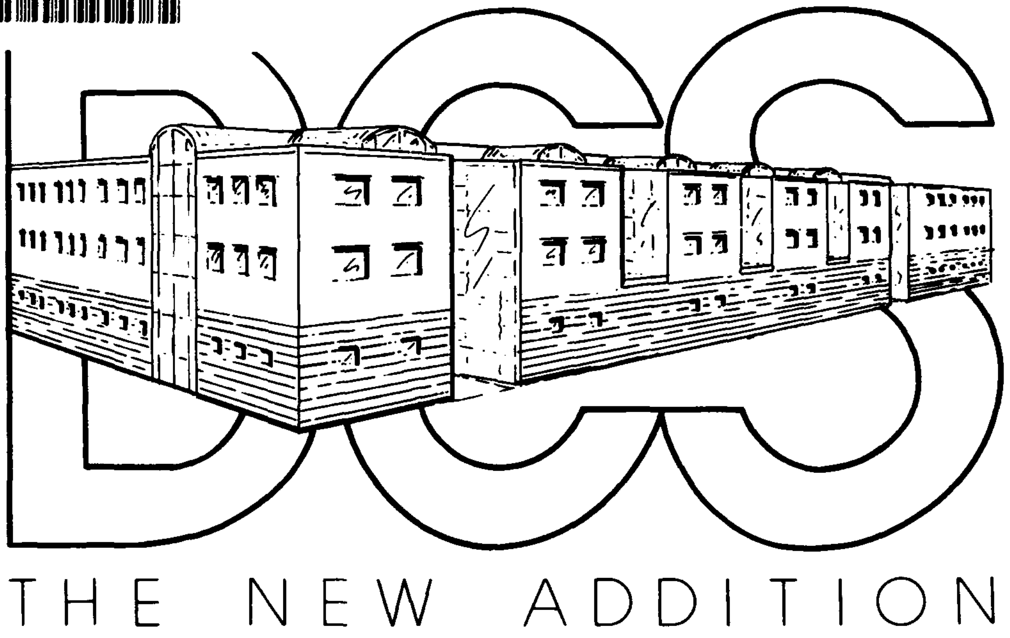
1

DEPARTMENT OF COMPUTER SCIENCE

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN



REPORT NO. UIUCDCS-R-91-1686

UILU-ENG-91-1732

LEARNING PROBABLY COMPLETABLE PLANS

by

Melinda T. Gervasio
Gerald F. DeJong

April 1991

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS None
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; Distribution unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) UIUCDCS-R-91-1686	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Dept. of Computer Science University of Illinois 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION Office of Naval Research
6c. ADDRESS (City, State, and ZIP Code) 1304 W. Springfield Urbana, Illinois 61801	7b. ADDRESS (City, State, and ZIP Code) 800 N. Quincy Street Arlington, VA 22217-5000
8a. NAME OF FUNDING/SPONSORING ORGANIZATION 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER N00014-86-K-0309
8c. ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)
Learning Probably Completable Plans

12. PERSONAL AUTHOR(S)
Melinda T. Gervasio and Gerald F. DeJong

13a. TYPE OF REPORT Technical	13b. TIME COVERED FROM ______ TO ______	14. DATE OF REPORT (Year, Month, Day) 1991 April	15. PAGE COUNT 15

16. SUPPLEMENTARY NOTATION

17. COSATI CODES

FIELD	GROUP	SUB-GROUP
05		10

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
Explanation-based learning, Incremental learning strategies
Disjunctive planning, Reactive planning

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

In completable planning, a planning system is given the ability to defer goals which it can prove to be achievable. This has the advantages of allowing the utilization of runtime information in planning and enabling a planner to use less precise a priori information without sacrificing guarantees of success. In this paper, we extend completable planning to goals which are only probably achievable, thus extending its scope to a wider variety of problems. We also define completable plans in terms of its constituent reactive plan components, conditionals and repeat-loops, which achieve the deferred goals, and we discuss the costs incurred by completable planning in terms of runtime evaluation cost, plan flexibility, a priori planning cost, and guarantees of success. In extending completable planning to probable achievability, we also introduce incremental explanation-based learning strategies for learning probably completable conditionals and probably completable repeat-loops, and demonstrate the learning of a probably completable plan in a simple train route-planning example.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT [✓ UNCLASSIFIED/UNLIMITED] ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL Alan Meyrowitz	22b. TELEPHONE (Include Area Code) (202)696-4302 22c. OFFICE SYMBOL

DD Form 1473, JUN 86 — Previous editions are obsolete. — SECURITY CLASSIFICATION OF THIS PAGE

Learning Probably Completable Plans*

Melinda T. Gervasio and Gerald F. DeJong

Beckman Institute For Advanced Science and Technology
University of Illinois at Urbana–Champaign
405 N. Mathews Ave., Urbana, IL 61801

(217) 244–1503
gervasio@cs.uiuc.edu dejong@cs.uiuc.edu

Main Topic: Machine Learning
Secondary Topics: Planning, Scheduling, and Reasoning About Action, Robotics and Control

ABSTRACT

In completable planning, a planning system is given the ability to defer goals which it can prove to be achievable. This has the advantages of allowing the utilization of runtime information in planning and enabling a planner to use less precise a priori information without sacrificing guarantees of success. In this paper, we extend completable planning to goals which are only probably achievable, thus extending its scope to a wider variety of problems. We also define completable plans in terms of its constituent reactive plan components, conditionals and repeat–loops, which achieve the deferred goals, and we discuss the costs incurred by completable planning in terms of runtime evaluation cost, plan flexibility, a priori planning cost, and guarantees of success. In extending completable planning to probable achievability, we also introduce incremental explanation–based learning strategies for learning probably completable conditionals and probably completable repeat–loops, and demonstrate the learning of a probably completable plan in a simple train route–planning example.

* This research was supported by the Office of Naval Research under grant N–0014–87–K–0309.

INTRODUCTION

Recent years have seen a growing interest in planning systems which have both a priori planning capabilities for providing goal–directedness, and reactive runtime capabilities for providing flexibility and sensitivity to the runtime environment [Drummond90, Gervasio90, Kaelbling88, Mitchell90, Turney89], in response impracticality of classical planning [Chapman87, Sacerdoti77, Sussman73], which constructs complete provably–correct plans but requires complete and correct a priori information to do so—an unrealistic demand in many real world domains. These same domains, however, although uncertain in some aspects, also often follow particular predictable patterns of behavior, and thus some a priori planning is both possible and desirable, in contrast to the unpredictable, dynamic environments addressed by situated action [Agre87, Suchman87].

In [Gervasio90], we presented an integrated planning approach wherein a classical planner is augmented with the ability to defer achievable goals, where *achievability* is simply defined as the existence of a plan which would achieve the goal during execution. If a planner could prove the existence of such a plan—not necessarily by determining the plan itself—the goal could be deferred until execution, when additional information could become available to make better–informed planning decisions. Furthermore, a system minimizes its a priori information requirements by being able to use less precise information. Also, because the deferred goals have achievability proofs, a system can still construct provably–correct plans. We also presented ***contingent explanation–based learning***, a strategy for learning general completable reactive plans, which introduced the idea of conjectured variables to distinguish between a priori and runtime planning decisions.

A limitation of this original approach to completable planning was the requirement of absolute achievability. Consider the problem of hammering a nail into the wall, where a pound action will often result in driving the nail further into the wall, but may sometimes end up getting the nail bent instead. A completable planner requiring absolute achievability would not be able to solve such a problem, because for the same knowledge limitations reason that it cannot determine a priori the precise number of pounding action to use, it cannot guarantee that the nail will not be bent by a pounding action. However, a completable planner which can reason about actions with different possible outcomes, and construct or learn alternative plans, can construct probably completable plans which have a good chance of success.

In this paper, we extend the completable planning approach to probable achievability. Introducing probable achievability not only extends the scope of completable planning to a wider variety of planning problems, but also opens up new opportunities to investigate the learning of completable plans. We begin by formalizing the idea of completable planning by defining the various components of a completable plan and the achievability constraints on these different components and discussing some of the cost tradeoffs involved in completable planning. We then present incremental learning strategies for learning probably completable plans and demonstrate its use in learning an increasingly completable plan for determining travel routes in a simple train domain. Finally, we discuss some related and future work.

COMPLETABLE PLANS

In completable planning, a planner may decide upon certain actions a priori and use runtime information gathered through its sensors to decide upon other actions during execution. All projection is done prior to execution, and runtime decision-making is limited to being *reactive*—i.e. during execution, the system decides on its next action on a predetermined basis, represented by conditionals and repeat-loops.[1] Uncertainty is characterized by using state descriptions which correspond to a set of states rather than unique, single states. Let a state description S be a conjunction of atomic sentences, and states(S) represent the set of all states in which S is satisfied. For a plan component p in a plan P, let PREC(p) be the state description resulting from regressing the goal back through all later plan components in P to p. Similarly, let EFF(p) be the result of projecting the initial state through all earlier plan components in P to p.

Given an initial state description I and goal state description G, a provably-correct *plan* P for I and G is an ordered sequence of plan components of the form: $\{p_1;p_2;...;p_n\}$, constrained in the following manner:

$\text{states}(I) \subseteq \text{states}(\text{PREC}(p_1))$

For $p_i \in \{p_1,p_2,...,p_{n-1}\}$, $\text{states}(\text{EFF}(p_i)) \subseteq \text{states}(\text{PREC}(p_{i+1}))$

$\text{states}(\text{EFF}(p_n)) \subseteq \text{states}(G)$.

This is shown graphically inFigure 1.

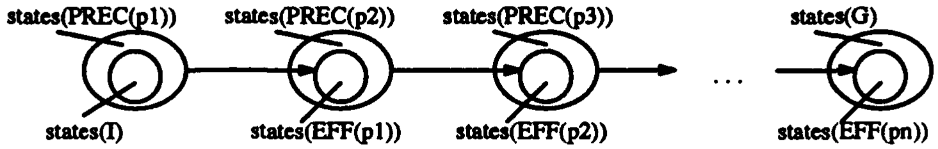


Figure 1. A plan consisting of unconditional actions.

There are three types of *plan components*: unconditional actions, conditionals, and repeat loops. *Unconditional actions* are actions to be executed without environmental input. Classical planners [Chapman87, Sacerdoti77, Sussman73] can be characterized as constructing plans consisting solely of unconditional actions, and thus unconditional actions can be said to constitute the classical part of a completable plan while conditionals and repeat-loops, the reactive part. In completable planning, the deferred goals addressed by the reactive components must be achievable, and thus constraints must be placed on conditionals and repeat-loops to guarantee their achievement of the preconditions of succeeding actions.

1. The decision to allow no further planning during execution is not a theoretical claim. The primary focus of this research is on learning, and as such, the system currently has simple planning capabilities. However, other planning capabilities may be added as the research progresses and new learning avenues are explored.

Probably Completable Conditionals

Conditionals deals with the problem of over–general initial state descriptions. Prior to execution, all a planner may know is that at a particular point in the execution of a plan, it will be in one of several possible states satisfying some description. However the different states satisfying this description may require different actions to achieve the preconditions for succeeding actions. For example, in planning to get to some higher floor in a new building, after going through the front door, all you may know is that you will be in the lobby. However, depending upon various factors such as whether there will be a staircase or an elevator or both, your proximity to each one, which floor you wish to go to, and the functionality of the elevator, you would like to take different actions. Through conditionals, decisions can be made during execution regarding appropriate actions, using any additional information which becomes available at that point in execution.

A *conditional* is of the form: {COND $c_1 \rightarrow q_1$; $c_2 \rightarrow q_2$; ...; $c_n \rightarrow q_n$} where each $c_i \rightarrow q_i$ is an *action–decision rule* which represents the decision to execute the plan q_i when the test c_i yields true. Like the situation–action type rules used in reactive systems such as [Kaelbling88, Mitchell90, Schoppers87], action–decision rules map different situations into different actions, allowing a system to make decisions based on its current environment.

In a completable plan, however, a conditional p_i = {COND $c_1 \rightarrow q_1$; $c_2 \rightarrow q_2$; ...; $c_n \rightarrow q_n$} must also satisfy the following constraints for achievability:

1. *Exhaustiveness*: states($c_1 \wedge c_2 \wedge ... \wedge c_n$) must be an exhaustive subset of states(EFF(p_{i-1}))
2. *Observability*: each c_i must consist of observable conditions, where an *observable condition* is one for which there exists a sensor which can verify the truth or falsity of the condition.
3. *Achievement*: for each q_i, states(EFF(q_i)) $\subseteq$ states(PREC(p_{i+1})).

This is shown graphically in Figure 2. For probably completable plans, the exhaus-

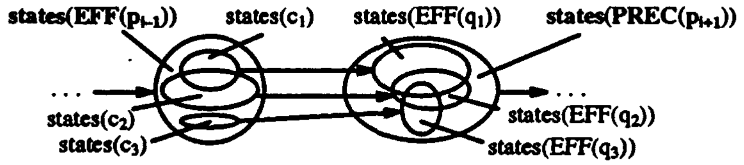


Figure 2. A completable conditional p_i with three action–decision rules.

tiveness constraint is relaxed to require only probable exhaustiveness, and the greater the coverage, the greater the conditional's chance of achieving PREC(p_{i+1}). The observability constraint requires knowledge of sensory capability, and here we use the term *sensor* in the broader sense of some set of sensory actions, which we will assume the system knows how to execute to verify the associated condition. It is needed to ensure that the conditional can be successfully evaluated during execution. Finally, the achievement constraint ensures that there the actions taken in the conditional achieve the preconditions of the succeeding plan component. Provided these three constraints are satisfied, the conditional is considered probably completable, and the goal PREC(p_{i+1}) of the conditional is probably achievable.

Probably Completable Repeat-Loops

A *repeat-loop* is of the form: {REPEAT q UNTIL c}, which represents the decision to execute the plan q until the test c yields true. Repeat loops are similar in idea to servo-mechanisms; but in addition to the simple yet powerful failure-recovery strategy such mechanisms provide, repeat loops also permit the construction of repeated action sequences achieving incremental progress towards the goal, which may be viewed as a reactive, runtime method of achieving generalization-to-N [Cohen88, Shavlik87]. Repeat loops are thus useful in completable plans for mainly two reasons: simple failure recovery and iterations for incremental progress.

Repeat-loops for simple failure-recovery are useful with actions having nondeterministic effects, which arise from knowledge limitations preventing a planner from knowing which of several possible effects a particular action will have. For example, in attempting to unlock the door to your apartment, driving the key to the keyhole will most often result in the key lodging into the hole. However, once in a while, the key may end up jamming beside the hole instead; but repeating the procedure often achieves the missed goal. In completable planning, if an action has several possible outcomes, and if the successful outcome is highly probable, and if the unsuccessful ones do not prevent the eventual achievement of the goal, then a repeat-loop can be used to ensure the achievement of the desired effects.

A repeat-loop p = {REPEAT q until c} for failure-recovery must satisfy the following constraints for achievability:

1. *Observability*: c must be an observable condition
2. *Achievement*: c must be a probable effect of q
3. *Repeatability*: the execution of q must not irrecoverably deny the preconditions of q until c is achieved.

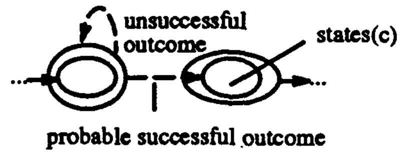
This is shown graphically in Figure 3a. The observability constraint is needed, once again, to be able to guarantee successful evaluation, while the achievement and repeatability constraints together ensure a high probability of eventually exiting the repeat loop with success. As with the exhaustiveness constraint for conditionals, the repeatability constraint may be relaxed so that the execution of q need only probably preserve or probably allow the reachievement of the preconditions of q.

Repeat-loops for incremental progress deal with over-general effect state description. Once again, knowledge limitations may result in a planner not having precise enough information to make action decisions a priori. In actions which result in changing the value of a quantity, for example, your knowledge may be limited to the direction of change or to a range of possible new values, which may not be specific enough to permit making decisions regarding precise actions—for example, determining the precise number of action repetitions or the precise length of time over which to run a process in order to achieve the goal. The implicit determination of such values during execution is achieved in completable planning through the use of repeat–loops which achieve incremental progress towards the goal and use run-time information to determine when the goal has been reached.

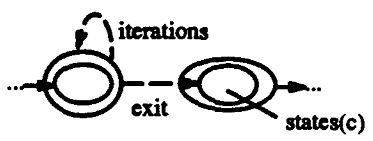
A repeat–loop p = {REPEAT c until p} for incremental progress must satisfy the following constraints for achievability:

1. *Continuous observability*: c must be an observable condition which checks a particular parameter for equality to a member of an ordered set of values—for example, a value within the range of acceptable values for a quantity.
2. *Incremental achievement*: each execution of q must result in incremental progress towards and eventually achieving c—i.e. it must reduce the difference between the previous parameter value and the desired parameter value by at least some finite non–infinitesimal ε.
3. *Repeatability*: the execution of q must not irrecoverably deny the preconditions of q until c is achieved.

This is shown graphically in Figure 3b. The continuous observability constraint en-



a. Failure recovery.



b. Incremental Progress.

Figure 3. Completable repeat–loops.

sures that the progress guaranteed by the incremental achievement and repeatability constraints can be detected and the goal eventually verified. For both failure recovery and interactions for incremental progress, if the repeat–loop satisfies the constraints, the repeat–loop is considered probably completable and the goal c is achievable.

Completable Plans vs. Universal Plans

A universal plan [Schoppers87] is the compilation of a planner's knowledge of actions with respect to a goal into a set of situation–action rules which provide a system with advice on what to do next in all possible situations, and may thus be viewed as the transformation of a set of classical plans into a purely reactive plan. A completable plan with conditionals is essentially derivable from a set of classical plans as well—i.e. a completable plan with a conditional may be viewed as a set of classical plans, one for each alternative. The primary difference is that in a universal plan, all the action decision are made during execution while in a completable plan, only particular planning decisions are left for execution.

For a particular planning problem, let U be a universal plan and C be a completable plan, where U corresponds to some complete set of classical plans P and C corresponds to some subset P′ of P. Assume that the procedure for evaluating both a universal plan and a conditional takes as input a set of rules and outputs one as the rule to be applied. Let:

r = *the average number of action–decision rules in a conditional P*
u = *the number of situation–action rules in U*
c_r = *the average cost of evaluating a rule*

Then rc_r is the cost of evaluating a conditional and uc_r is the cost of evaluating the situation–action rules. In the worst case, C will itself be equivalent to a universal plan for P′, with every action determined by a conditional containing all the situation–action rules for P′, but since $P' \subseteq P$, $rc_r < uc_r$. Furthermore, let:

n = *the average number of actions taken using U to achieve the goal*
d = *the number of deferred decisions—i.e. conditionals—in C.*

Since every action decision in U is made by evaluating the rules of U, the total runtime evaluation cost of U is nuc_r, while that of P is drc_r. Since in P some action decisions will generally be made a priori, d will generally be less than n, and thus, $drc_r < nuc_r$. Intuitively, it seems the size r of a conditional would be much less than the size u of the universal plan, since a conditional is a very restricted set of action–decision rules for achieving a particular goal, whereas a universal plan encompasses all the intermediate goals and states, and thus probably $drc_r << nuc_r$. For the same reason, however, U is much more flexible than C. Furthermore, since U does not check any preconditions prior to execution, if we let

a = *the number of (a priori) preconditions in P*
c_p = *the average cost of verifying a precondition*

then C incurs an additional cost ac_p over U for verifying plan applicability. However, this cost comes with the benefit of a completable planner being able to determine, before any actions are executed, whether the plan will achieve the goal. In the imperfectly–characterizable but fairly well–behaved domains for which completable planning is designed, trading off flexibility for guarantees of success is probably an advantageous decision. Current work includes designing experiments to investigate this tradeoff as well as the actual costs incurred, and the relative benefits and disadvantages brought by completable planning.

LEARNING PROBABLY COMPLETABLE PLANS

The planning problem has provided a wealth of research opportunities for the learning community, as evidenced by work such as [Chien89, DeJong89, Hammond86, Minton85, Mitchell90, Mooney88]. In [Gervasio90], explanation–based learning [DeJong86, Mitchell86] was shown to be useful in learning completable plans which involved the deferment of determining the length of time over which to let a process run to achieve a particular value for a continuously–changing quantity. Such a deferred decision may be represented in the formalization presented in the previous section by a repeat–loop iterating over a wait or a no–op, with the exit condition c testing for the goal value. Repeat–loops for incremental progress can thus be learned by constructing explanations about how the general behavior of repeated actions guarantees incremental progress towards the goal. Here, we present an incremental strategy for learning conditionals and repeat–loops for simple failure–recovery in probably completable plans.

The idea of probably completable plans lends itself naturally to incremental learning strategies. Conditionals, for example, represent a partitioning of a set of states into subsets requiring different actions to achieve the same goal. With probable achievability, a plan may include only some of these subsets. As problems involving the excluded subsets are encountered, however, the plan can be modified to include the new conditions and actions. Similarly, incremental learning can be used to learn failure–recovery strategies within repeat–loops. The motivation behind the incremental learning of reactive components is similar to the motivation behind much work on approximations and learning from failure, including [Bennett90, Chien89, Hammond86, Mostow87, Tadepalli89]. The primary difference between these approaches and completable planning is that in these approaches, a system has the ability to correct the assumptions behind its incorrect approximations and thus tends to converge upon a single correct solution for a problem. In completable planning, uncertainty is inherent in the knowledge representation itself and the system instead addresses the problem of ambiguity through reactivity. As a system learns improved reactive components, it thus tends to increase a plan's coverage of the possible states which may be reached during execution.

Learning Probably Completable Conditionals

Since preconditions for the actions in an action–decision rule may be satisfied either through initial state information or through the conditions in the rule, a general plan learned through EBL[2] must be further processed to distinguish between these two types of preconditions:

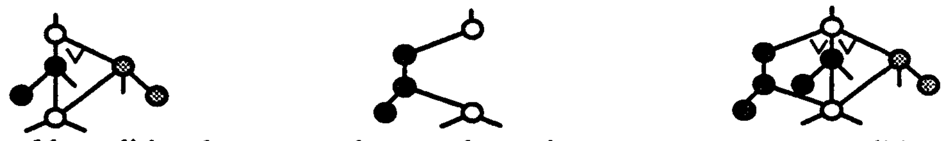
```
For each precondition pr
    If pr is not satisfied by I
    then If pr is observable
        then  Find all operators supported by pr
              Make the execution of that operator conditional on pr
              Remove pr from the general plan's preconditions.[3]
```

Recall that for conditionals to be completable, they must satisfy the constraints of exhaustiveness, observability, and achievement. Since the plans here are derived from explanations, the constraint of achievement is already satisfied. The procedure above checks for observability. For the exhaustiveness constraint, let X be the desired minimum coverage, where X can be a user–supplied value or one computed from other parameters such as available resources and importance of success. Coverage can be represented by probabilities, either qualitative or quantitative, in our case, qualitative (i.e. using qualitative terms such as "usually" to denote high probability). Then the exhaustiveness constraint is satisfied in a conditional $\{\text{COND } c_1 \rightarrow q_1; \ldots ; c_n \rightarrow q_n\}$ iff the probability of $(c_i \vee c_2 \vee \ldots \vee c_n)$ is at least X.

2. EBL is used in completable planning to learn macro–operators or general plans. The planner then simply looks for a single applicable general plan when given a planning problem. No chaining on macro–operators is performed.

3. The distinction between initial state information and runtime information is an important one in completable planning, and it is assumed that the learning system keeps track of or is able to reason about what information was available initially and what came in during execution.

A conditional manifests itself in an explanation a multiple, disjunctive paths between two nodes (Figure 4a), with a path representing one action–decision rule, its leaves which cannot be satisfied in the initial state forming the condition and the operators along the path forming the action.[4] Since coverage may be incomplete, a system may at one time fail to satisfy any of the conditions within a conditional, in which case, the system has the option of learning a new alternative (Figure 4b) to solve the current problem and to increase coverage in future problems (Figure 4c). Merging



a. old conditional b. new alternative c. new conditional

Figure 4. Explanation Structures in learning new conditionals.

a new rule into a conditional can be done using the old plan and a plan with the new alternative as follows:[5]

```
new–to–add := plan components in new plan not matching any in old plan
old–to–change := plan component in old plan not matching any in new plan
Make a new action–decision rule using new–to–add
Append the new rule to the action–decision rules of old–to–change
For each precondition pr in the new plan
    If pr is not already in the old plan
    then add pr to the preconditions of the old plan.
```

Learning Probably Completable Repeat–Loops

Repeat–loops for simple failure–recovery address the problem of actions with nondeterministic effects or multiple possible outcomes, and thus repeat–loops are first constructed by identifying such actions in the general plan:

```
For each action a in the plan
    If the outcome of a used in the plan is a probable outcome among others
    then If the desired outcome c is observable
        then Construct a repeat loop for a.
```

Recall that for a repeat–loop for failure to be completable, it must satisfy the constraint of repeatability aside from the constraints of observability and achievement. If the unsuccessful outcomes of a do not prevent the repetition of a, then the repeatability constraint is satisfied, and the probable eventual achievement of the desired effects is guaranteed. However, for unsuccessful outcomes which deny the preconditions to a, actions to recover the preconditions must be learned. These precondition–recovery strategies within a repeat–loop can be characterized as a conditional, where

4. Initially, a plan may contain a conditional containing only one action–decision rule, a case arising when there is only one known action but some of whose actions need to be verified at execution.

5. Plans (their explanations) are associated with specific and general bindings (see [Mooney86] for more on bindings). This procedure uses the specific bindings of both plans to determine equality and as equal components and preconditions are found, the combined general bindings are updated to effectively achieve the merging, with the final general bindings are used in the modified old plan.

the different states are the different outcomes, the different actions are the different precondition–recovery strategies, and the common effect state is the precondition state of the action a. If we let u_i be an unsuccessful outcome, and r_i be the recovery strategy for u_i, then a repeat–loop eventually takes the form {REPEAT {q; [COND $u_1 \rightarrow r_1$; ... ; $u_n \rightarrow r_n$]} UNTIL c}. Learning the embedded conditional for failure recovery can be done as in the previous section.

Example

A completable planning system implemented in Common LISP on an IBM RT Model 125 was given the task of learning a plan to get from one small city to another going through two larger cities using a train. The primary source of incompleteness preventing complete a priori planning is the system's knowledge with regard to the state of the railroads. In order for a system to get from one city to another, the cities have to be connected by a railroad, and the railroad has to be clear. For a railroad to be considered clear, it must be not flooded, not be congested with traffic, be free of accidents, and not be under construction. These conditions cannot always be verified a priori for all railroads, hence the need for conditionals.

The training example involves getting from the city of Wyne to the city of Ruraly, where the rail connectivity between the two cities is shown in Figure 5. Here, the railroad A–B is a major railroad and sometimes gets congested. Also, the northern railroads to and from X, C, and Z are susceptible to flooding. And accidents and construction may occur from time to time.[6]

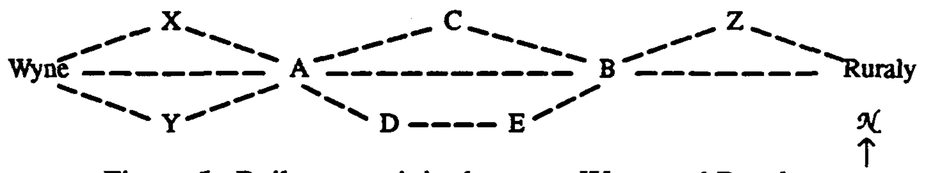


Figure 5. Rail connectivity between Wyne and Ruraly.

Learning Initial Plan. The initial training example given to the system is the route Wyne–A–B–Ruraly, since this is generally the quickest way to get from Wyne to Ruraly.[7] The initial derived plan is processed to determine conditionals and the learned general plan is shown in Figure 6. This is a plan for getting from one city to another

6. In the learning instances presented here, only the Wyne–to–Ruraly problem is used. This is partly to simplify the presentation and partly to avoid confounding the problem of learning conditionals with the generalization–to–N problem [Cohen88, Shavlik87], of which route–planning between several cities can be seen as an instance. One drawback of this decision is that the issue of evaluating conditionals can be sidestepped for the meantime; by presenting the system with alternatives in the order of their desirability, the action–decision rules in the resulting conditional can simply be evaluated in sequence until one applies. A possible non–implementation solution would be to let all the action–decision rules fire and then to use some metric (for example, route–length) to decide between the applicable alternatives.

7. The optimization problem is a very interesting research problem in itself, but beyond the scope of the current research. Here, we assume that the expert provides already optimized solutions as training examples.

```
PLAN 1
[COMPS
 [COND ((NOT (ACC ?RT22692)) (NOT (CONSTR ?RT22692)))
          -> ((GO ?AGT22688 ?CITY122689 ?CITY222690 ?RT22692))
 [COND ((NOT (ACC ?RT22697)) (NOT (CONSTR ?RT22697)) (NOT (TRAFF ?RT22697)))
          -> ((GO ?AGT22688 ?CITY222690 ?CITY222695 ?RT22697))
 [COND ((NOT (ACC ?RT22702)) (NOT (CONSTR ?RT22702)))
          -> ((GO ?AGT22688 ?CITY222695 ?CITY222700 ?RT22702))
[PRECS (AT ?AGT22688 ?CITY122689) (CONN ?CITY122689 ?CITY222690 ?RT22692) (NOT (TRAFF ?RT22692))
  (NOT (FLOOD ?RT22692)) (CONN ?CITY222690 ?CITY222695 ?RT22697) (NOT (FLOOD ?RT22697)) (CONN
  ?CITY222695 ?CITY222700 ?RT22702) (NOT (TRAFF ?RT22702)) (NOT (FLOOD ?RT22702))
[EFFS (AT ?AGT22688 ?CITY222700)]
[EXPL: [EXPLANATION for (AT AMATRAK RURALY)]]
```

Figure 6. Initial Learned Plan.[8]

with two intermediate stops, where only the railroad between the two intermediate cities is susceptible to heavy traffic and must thus be checked for it.

Learning Alternative Plans. When the system encounters a situation in which none of the conditions in a conditional is satisfied—in this example, (not (has–heavy–traffic A–B)) proves false just as the system is to execute (go Amatrak A B A–B) to achieve (at Amatrak B)—the system needs to learn a new route from A to B in order to get back on track. The solution given to the system is the route A–C–B, which gets the system to B and allows it to continue with the next step in its original plan and achieve its goal of getting to Ruraly. From this experience, the system modifies its old plan to include the new alternative of going through another city between the two intermediate cities. The system thus now has two alternatives when it gets to city A. When it encounters a situation in which A–B is congested and A–C is flooded, it is given another alternative solution, A–D–E–B, from which it learns another plan to get from A to B and modifies the old plan as before. Now, in planning to get from Wyne to Ruraly, the system is able to construct the plan in Figure 7.

```
PLAN 1
[COMPS
 [COND ((NOT (ACC WYNE-A)) (NOT (CONSTR WYNE-A))) -> ((GO AMATRAK WYNE A WYNE-A))]
 [COND  ((NOT (ACC A-B)) (NOT (CONSTR A-B)) (NOT (TRAFF  A-B)) -> ((GO AMATRAK A B A-B))
         ((NOT (ACC A-C)) (NOT (CONSTR A-C)) (NOT (FLOOD A-C)))
           -> (((GO AMATRAK A C A-C))
               (COND (((NOT (ACC C-B)) (NOT (CONSTR C-B)) (NOT (FLOOD C-B)))
                         -> ((GO AMATRAK C B C-B)))))
         ((NOT (ACC A-D)) (NOT (CONSTR A-D)))
           -> (((GO AMATRAK A D A-D))
               (COND (((NOT (ACC D-E)) (NOT (CONSTR D-E))) -> ((GO AMATRAK D E D-E))))
               (COND (((NOT (ACC E-B)) (NOT (CONSTR E-B))) -> ((GO AMATRAK E B E-B))))))]
 [COND ((NOT (ACC B-RURALY)) (NOT (CONSTR B-RURALY)))
          -> ((GO AMATRAK B RURALY B-RURALY))]
[PRECS (AT AMATRAK WYNE) (CONN WYNE A WYNE-A) (NOT (TRAFF WYNE-A)) (NOT (FLOOD
  WYNE-A)) (CONN A B A-B) (NOT (FLOOD A-B)) (CONN B RURALY B-RURALY) (NOT (TRAFF
  B-RURALY)) (NOT (FLOOD B-RURALY)) (CONN A C A-C) (NOT (TRAFF A-C)) (CONN C B C-B) (NOT
  (TRAFF C-B)) (CONN A D A-D) (NOT (TRAFF A-D)) (NOT (FLOOD A-D)) (CONN D E D-E) (NOT (TRAFF D-E)) (NOT
  (FLOOD D-E)) (CONN E B E-B) (NOT (TRAFF E-B)) (NOT (FLOOD E-B))
[EFFS (AT AMATRAK RURALY)]
[EXPL: [EXPLANATION for (AT AMATRAK RURALY)]]
```

Figure 7. Final specific plan for getting to Ruraly from Wyne.[9]

8. For brevity, the following abbreviations have been used: conn for connected, acc for has–accident, constr for under–construction, traff for has–heavy–traffic, and flood for flooded.

9. Portions added by new plans shown in two sets of italics.

DISCUSSION & CONCLUSIONS

Note that there are usually many possible alternative plans—in the railroad-route-planning problem, for example, there are as many train routes as there are railroads between any two cities. Unless there is reason to learn new alternatives, however, effort will not be expended in learning these alternatives. This minimizes the execution-time cost of evaluating conditionals by keeping conditionals small, as well as the cost of checking preconditions, since new action-decision rules also usually add preconditions to be checked for plan applicability.

A direction for future work is a more thorough analysis of the tradeoff between the advantages brought and costs incurred by completable planning. Aside from the a priori planning cost completable plans have over reactive plans, and the runtime evaluation cost completable plans have over classical plans, in proving achievability completable plans also sometimes require knowledge about the general behavior of actions not always available in traditional action definitions. On the other hand, completable planning also minimizes a priori information requirements. Another direction for future work is in integrating probabilities more fully into the completable planning framework. This would involve using qualitative probabilities (as in [Wellman90]) or quantitative probabilities (as in [Hanks90]) and characterizing their relationship to achievability, which may help quantify the notion of achievability as well as open up new areas in which to explore learning.

[Martin90] presents a planning approach wherein an a priori *strategic planner* defers to the *reactive planner* those planning decisions the reactive planner has proven (through experience) itself capable of handling. In contrast, the achievability criterion used in our work is knowledge-based, rather than empirical, although a combination of both is currently being investigated. The conditionals in this work are also related to the work on disjunctive plans, such as [Fox, Mello86], however these have been focused more towards the construction of complete, flexible plans for closed-world manufacturing applications, whereas the incremental learning strategy presented here was designed precisely for problems where accounting for all contingencies is expected to be intractable. The idea of incrementally improving a plan's coverage has also been investigated in [Drummond90], where a plan's chance of achieving the goal is increased through *robustification*, the gradual consideration of other possible outcomes of actions and construction of failure-recovery strategies for them. Here, aside actions with different possible outcomes, we deal with the problem of over-general knowledge. And as discussed earlier, there is also much related work on learning good approximations in planning, including [Bennett90, Chien89, Hammond86, Mostow87, Tadepalli89].

In this paper, we have extended the idea of completable planning by allowing an a priori planner to defer goals which are only probably achievable. We defined completable planning in terms of its plan components, as well as the achievability constraints on conditionals and repeat-loops, and we discussed various cost tradeoffs. Finally, we presented and demonstrated incremental strategies for learning conditionals and repeat-loops in completable plans.

References

[Agre87] P. Agre and D. Chapman, "Pengi: An Implementation of a Theory of Activity," *Proceedings of the National Conference on Artificial Intelligence*, Seattle, WA, July 1987, pp. 268–272.

[Bennett90] S. W. Bennett, "Reducing Real–world Failures of Approximate Explanation–based Rules," *Proceedings of the Seventh International Conference on Machine Learning*, Austin, TX, June 1990, pp. 226–234.

[Chapman87] D. Chapman, "Planning for Conjunctive Goals," *Artificial Intelligence 32*, 3 (1987), pp. 333–378.

[Chien89] S. A. Chien, "Using and Refining Simplifications: Explanation–based Learning of Plans in Intractable Domains," *Proceedings of The Eleventh International Joint Conference on Artificial Intelligence*, Detroit, MI, August 1989, pp. 590–595.

[Cohen88] W. W. Cohen, "Generalizing Number and Learning from Multiple Examples in Explanation Based Learning," *Proceedings of the Fifth International Conference on Machine Learning*, Ann Arbor, MI, June 1988, pp. 256–269.

[DeJong86] G. F. DeJong and R. J. Mooney, "Explanation–Based Learning: An Alternative View," *Machine Learning 1*, 2 (April 1986), pp. 145–176.

[DeJong89] G. F. DeJong, "Explanation–Based Learning with Plausible Inferencing," *Proceedings of The Fourth European Working Session on Learning*, Montpellier, Dec. 1989, pp. 1–10.

[Drummond90] M. Drummond and J. Bresina, "Anytime Synthetic Projection: Maximizing the Probability of Goal Satisfaction," *Proceedings of the Eighth National Conference on Artificial Intelligence*, Boston, MA, August 1990, pp. 138–144.

[Fox] B. R. Fox and K. G. Kempf, "Opportunistic Scheduling for Robotic Assembly," *Proceedings of the 1985 Institute of Electrical and Electronics Engineers International Conference on Robotics and Automation*,, pp. 880–889.

[Gervasio90] M. T. Gervasio, "Learning General Completable Reactive Plans," *Proceedings of the Eighth National Conference on Artificial Intelligence*, Boston, MA, August 1990, pp. 1016–1021.

[Hammond86] K. Hammond, "CHEF: A Model of Case–Based Planning," *Proceedings of the National Conference on Artificial Intelligence*, Philadelphia, PA, August 1986, pp. 267–271.

[Hanks90] S. Hanks, "Practical Temporal Projection," *Proceedings of the Eighth National Conference on Artificial Intelligence*, Boston, MA, August 1990, pp. 158–163.

[Kaelbling88] L. P. Kaelbling, "Goals as Parallel Program Specifications," *Proceedings of The Seventh National Conference on Artificial Intelligence*, Saint Paul, MN, August 1988, pp. 60–65.

[Martin90] N. G. Martin and J. F. Allen, "Combining Reactive and Strategic Planning through Decomposition Abstraction," *Proceedings of the Workshop on Innovative Approaches to Planning, Scheduling and Control*, San Diego, CA, November 1990, pp. 137–143.

[Mello86] L. S. H. Mello and A. C. Sanderson, "And/Or Graph Representation of Assembly Plans," *Proceedings of the National Conference on Artificial Intelligence*, Philadelphia, PA, August 1986, pp. 1113–1119.

[Minton85] S. Minton, "Selectively Generalizing Plans for Problem–Solving," *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, August 1985, pp. 596–599.

[Mitchell86] T. M. Mitchell, R. Keller and S. Kedar–Cabelli, "Explanation–Based Generalization: A Unifying View," *Machine Learning 1*, 1 (January 1986), pp. 47–80.

[Mitchell90] T. M. Mitchell, "Becoming Increasingly Reactive," *Proceedings of the Eighth National Conference on Artificial Intelligence*, Boston, MA, August 1990, pp. 1051–1058.

[Mooney86] R. J. Mooney and S. W. Bennett, "A Domain Independent Explanation-Based Generalizer," *Proceedings of the National Conference on Artificial Intelligence*, Philadelphia, PA, August 1986, pp. 551–555.

[Mooney88] R. J. Mooney, "Generalizing the Order of Operators in Macro-Operators," *Proceedings of the Fifth International Conference on Machine Learning*, Ann Arbor, MI, June 1988, pp. 270–283.

[Mostow87] J. Mostow and N. Bhatnager, "Failsafe – A Floor Planner that Uses EBG to Learn from its Failures," *Proceedings of the Tenth International Conference on Artificial Intelligence*, Milan, Italy, August 1987.

[Sacerdoti77] E. Sacerdoti, *A Structure for Plans and Behavior*, American Elsevier, New York, 1977.

[Schoppers87] M. J. Schoppers, "Universal Plans for Reactive Robots in Unpredictable Environments," *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, Milan, Italy, August 1987, pp. 1039–1046.

[Shavlik87] J. W. Shavlik and G. F. DeJong, "An Explanation-Based Approach to Generalizing Number," *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, Milan, Italy, August 1987, pp. 236–238.

[Suchman87] L. A. Suchman, *Plans and Situated Actions*, Cambridge University Press, Cambridge, 1987.

[Sussman73] G. J. Sussman, "A Computational Model of Skill Acquisition," Technical Report 297, MIT AI Lab, Cambridge, MA, 1973.

[Tadepalli89] P. Tadepalli, "Lazy Explanation-Based Learning: A Solution to the Intractable Theory Problem," *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, Detroit, MI, August 1989.

[Turney89] J. Turney and A. Segre, "SEPIA: An Experiment in Integrated Planning and Improvisation," *Proceedings of The American Association for Artificial Intelligence Spring Symposium on Planning and Search*, March 1989, pp. 59–63.

[Wellman90] M. P. Wellman, "The STRIPS Assumption for Planning Under Uncertainty," *Proceedings of the Eighth National Conference on Artificial Intelligence*, Boston, MA, August 1990, pp. 198–203.